Project 2

Daniel Fredin, Junhan Li, & Eric Chen

Problem 1

Problem 3

a)

ii) 10-fold Cross-Validation

```
# 10-fold Cross-validation method
ctrlspecs <- trainControl(method="cv",
                          number=10,
                          savePredictions="all",
                          classProbs=TRUE)

set.seed(1234)
cvmodel <- train(COMMIT ~ .,
                 data=train,
                 method="glm",
                 family = "binomial",
                 trControl=ctrlspecs)
summary(cvmodel)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.06170  -0.90585   0.08378   0.93637   1.94307
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -10.38779    5.07406  -2.047   0.0406 *
## AGE           0.14086    0.09033   1.559   0.1189
## SALARY        0.11065    0.09566   1.157   0.2474
## CLASSSIZE    -0.14105    0.06897  -2.045   0.0408 *
## RESOURCES     0.18733    0.13923   1.345   0.1785
## AUTONOMY      0.30463    0.12547   2.428   0.0152 *
## CLIMATE       0.21419    0.09829   2.179   0.0293 *
## SUPPORT      -0.19250    0.13418  -1.435   0.1514
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 166.36  on 119  degrees of freedom
## Residual deviance: 130.47  on 112  degrees of freedom
## AIC: 146.47
##
## Number of Fisher Scoring iterations: 4
```

```
# results1 <- data.frame( R2 = R2(predictions, test$COMMIT),
#              RMSE = RMSE(predictions, test$COMMIT),
#              MAE = MAE(predictions, test$COMMIT))
# # Results for 10-fold Cross-validation method
# results1

# Model for 10-fold Cross-validation method
print(cvmodel)
```

```
## Generalized Linear Model
##
## 120 samples
##   7 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 108, 108, 108, 108, 108, 108, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.675     0.35
```

```
varImp(cvmodel)
```

```
## glm variable importance
##
##           Overall
## AUTONOMY   100.00
## CLIMATE     80.43
## CLASSSIZE   69.90
## AGE         31.67
## SUPPORT     21.86
## RESOURCES   14.85
## SALARY       0.00
```

```
# Predict outcome using model from training data based on testing data
predictions <- predict(cvmodel, newdata=test)

# Create confusion matrix to assess model fit/performance on test data
con_matx <- confusionMatrix(data=predictions, test$COMMIT)
con_matx
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction no yes
##        no  13   4
##        yes  2  11
##
##                Accuracy : 0.8
##                  95% CI : (0.6143, 0.9229)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.0007155
##
##                   Kappa : 0.6
##
##  Mcnemar's Test P-Value : 0.6830914
##
##             Sensitivity : 0.8667
##             Specificity : 0.7333
##          Pos Pred Value : 0.7647
##          Neg Pred Value : 0.8462
##              Prevalence : 0.5000
##          Detection Rate : 0.4333
##    Detection Prevalence : 0.5667
##       Balanced Accuracy : 0.8000
##
##        'Positive' Class : no
##
```

b)

c)

```
compute_weibull_stats <- function(shape, scale) {
  # Compute the mean
  mean_value <- scale * gamma(1 + (1/shape))


  # # Compute the median
  # median_value <- scale * qweibull(0.5, shape, scale)


  # Compute the median
  median_value <- scale * (log(2)^(1/shape))


  # Compute the mode
  if (shape > 1)
    mode_value <- scale * ((shape - 1) / shape)^(1/shape)
  else
    mode_value <- 0

  # Compute the variance
  variance_value <- scale^2 * (gamma(1 + (2/shape)) - (gamma(1 + (1/shape)))^2)

  # Return the computed statistics as a named list
  return(list(mean = mean_value, median = median_value, mode = mode_value, variance = variance_value))
}

# lambda = scale
# K = shape

# Example usage
scale_param <- 6
shape_param <- 1

stats <- compute_weibull_stats(shape_param, scale_param)

# Accessing the computed statistics
mean_value <- stats$mean
median_value <- stats$median
mode_value <- stats$mode
variance_value <- stats$variance

# Printing the computed statistics
cat("Mean:", mean_value, "\n")
```

```
## Mean: 6
```

```
cat("Median:", median_value, "\n")
```

```
## Median: 4.158883
```

```
cat("Mode:", mode_value, "\n")
```

```
## Mode: 0
```

```
cat("Variance:", variance_value, "\n")
```

```
## Variance: 36
```